Amendments to the Claims

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (currently amended) An interpreter performing operations specified in a computer program ~~consisting~~ comprised of instructions, said instructions being translated into an intermediate format comprising an intermediate code for said instructions, using a service routine to perform the semantics of an instruction, comprising:

~~statistics~~ means for collecting and recording statistics of how often service routines are executed and what parameters they had;

~~clustering~~ means for grouping ~~(SR89, SR17 . . . SR6; SR4, SR34 . . . SR16)~~ frequently used service routines with program jumps between each other in a program function with regard to a predetermined frequency value for determining such service routines;

~~said statistics~~ means for recording the frequency of service routines executed after ~~a~~ said function; and

~~encoding~~ means for assigning ~~a~~ frequently used service routines a shorter code than service routines ~~(SR3, SR57, SR94 . . . SR64)~~ executed after ~~a~~ said function~~, thus gathered statistics,~~ from an execution~~, control an encoding to optimize frequently used service routines for faster execution speed~~.

2. (currently amended) An interpreter according to claim 1, wherein more than one program function is grouped by grouping (SR89, SR17 . . . SR6; SR4, SR34 . . . SR16) frequently used service routines with jumps between each other in the same function~~, thus minimizing jumps between functions~~.

3. (currently amended) An interpreter according to claim 1, wherein said ~~encoding~~ code assigning means reduces usage of ~~provides for a reduced bandwidth to an~~ electronic memory bandwith when fetching intermediate code~~, as a frequently executed instruction is shorter~~.

4. (currently amended) An interpreter according to claim 1, further comprising a simulator, and wherein the interpreter is configured to collect said statistics ~~is collected before a~~before compilation of said simulator~~ is compiled~~.

5. (currently amended) An interpreter according to claim 1, further comprising a simulator, and wherein the interpreter is configured to collect said statistics~~ is collected~~ while ~~a~~ said simulator is running such that said~~, in which case the~~ simulator dynamically updates ~~a~~ said function with service routines used to simulate an instruction set.

6. (currently amended) An interpreter according to claim 1, wherein the ~~statistics and the encoding provides that a realistic~~ instruction sets which ~~is~~ are translated into service routines ~~can be~~are written in a high-level programming language~~ for a specific simulator task~~.

7. (currently amended) An interpreter according to claim 6, wherein the interpreter is ~~provided to be~~ written in ~~the~~ standard ISO C~~ through said statistics and encoding~~.

8. (currently amended) An interpreter according to claim 1, wherein said program function has a plurality of service routines grouped in it, and wherein the interpreter is further configured to share a program branch with every one of said service routines in said program function~~ to a next service routine may be shared by all service routines in a function through said statistics and encoding~~.

9. (currently amended) An interpreter according to claim 8, wherein the interpreter is further configured to employ a branch prediction table that ~~it~~ reduces the number of jumps between program functions ~~that are hard to predict in a branch prediction table, causing a table to functioning better on current processor architectures~~.

10. (currently amended) An interpreter according to claim 1, wherein the interpreter is further configured with profile driven compilation ~~is used to further enhance a simulator performance through said statistics and encoding~~.

11. (currently amended) An interpreter according to claim 1, ~~further providing for~~wherein the interpreter is further configured to perform compiler register mapping of often used variables~~, as the variables may be allocated as~~ into local variables.

12. (currently amended) An interpreter according to claim 1, wherein the interpreter is configured with no ~~it provides for avoiding use of~~ compiler specific extensions~~, thus providing for compiler independence~~.

13. (currently amended) An interpreter according to claim 1, wherein the interpreter is configured to place said codes assigned to said frequently executed service routines in said program function in a sequential block improving ~~it improves~~ instruction cache performance ~~by placing frequent code in a sequential block due to a common function for service routines~~.

14. (currently amended) An interpreter according to claim 1, further comprising an emulator and wherein ~~it~~the interpreter is used by ~~an~~said emulator.

15. (currently amended) A method for an interpreter performing operations specified in a computer program ~~consisting~~ comprised of instructions and~~, said instructions being translated to an intermediate format comprising an intermediate code for said instructions,~~ using ~~a~~ service routines to perform ~~the semantics of an instruction~~the instructions, comprising the steps of:

(a) dynamically collecting and recording statistics of how often service routines are executed and what parameters they had;

(b) grouping ~~(SR89, SR17 . . . SR6; SR4, SR34 . . . SR16)~~ frequently used service routines in a program function based on a predetermined frequency value for determining such service routines with program jumps between each other in said~~a~~ program function ~~with regard to a predetermined frequency value for determining such service routines~~;

(c) ~~said statistics~~ dynamically collecting and recording statistics of the frequency of service routines executed after ~~a~~ said program function; and

(d) ~~encoding for~~ assigning frequently used service routines a shorter code than service routines ~~(SR3, SR57, SR94 . . . SR64)~~ executed after ~~a~~ said program function~~, thus gathered statistics, from an execution, control an encoding to optimize frequently used service routines for faster execution speed~~.

16. (currently amended) A method according to claim 15, wherein more than one said program function is ~~grouped~~ produced by grouping ~~(SR89, SR17 . . . SR6; SR4, SR34 . . . SR16)~~ frequently used service routines in step (b)~~ with jumps between each other in the same function, thus minimizing jumps between functions~~.

17. (currently amended) A method according to claim 15, further comprising the step of ~~wherein said encoding provides for a reduced bandwidth to~~fetching an intermediate code from an electronic memory~~ when fetching intermediate code, as a frequently executed instruction is shorter~~.

18. (currently amended) A method according to claim 15, providing a simulator and wherein said steps (a) and (c) are performed before ~~statistics is collected before a~~ said simulator is compiled.

19. (currently amended) A method according to claim 15, providing a simulator and wherein steps (a) and (c) are performed ~~said statistics is collected~~ while ~~a~~ said simulator is running dynamically updating said function with service routines used to simulate an instruction set~~, in which case the simulator dynamically updates a function with service routines used to simulate an instruction set~~.

20. (currently amended) A method according to claim 15, wherein ~~the statistics and the encoding provides that a realistic instruction set~~said instructions which are~~is~~ translated into service routines are ~~can be~~ written in a high-level programming language ~~for a specific simulator task~~.

21. (currently amended) A method according to claim 20, wherein the interpreter is ~~provided to be~~ written in a ~~the standard ISO~~ C programming language ~~through said statistics and encoding~~.

22. (currently amended) A method according to claims 15, wherein a program branch to a next service routine ~~may be~~is shared by all service routines in ~~a~~ said function ~~through said statistics and encoding~~.

23. (currently amended) A method according to claim 22, the step further comprising using ~~wherein it reduces the number of jumps, which are hard to predict, in~~ a branch prediction table~~, causing a table to functioning better on current processor architectures performance through said statistics and encoding~~.

24. (currently amended) A method according to claim 15, ~~wherein~~ the step further comprising using profile driven compilation ~~is used to further enhance a simulator performance through said statistics and encoding~~.

25. (currently amended) A method according to claim 15, ~~wherein it provides for~~ the step further comprising using compiler register mapping of often used variables~~, as the variables may be allocated as local variables performance through said statistics and encoding~~.

26. (currently amended) A method according to claim 15, wherein ~~it provides for avoiding use of~~ no compiler specific extensions are used~~, thus providing for compiler independence performance through said statistics and encoding~~.

27. (currently amended) A method according to claim 15, wherein said codes assigned in step (d) of said frequently used service routines of said program function are placed in a sequential block improving ~~it improves~~ instruction cache performance ~~by placing frequent code in a sequential block due to a common function for service routines~~.

28. (currently amended) A method according to claim 15, providing an emulator that uses the interpreter ~~wherein it is used by an emulator~~.

29. (currently amended) An interpreter performing operations of computer program code instructions by means of service routines, comprising:

a statistics mechanism devised to register the frequency of execution and the execution parameters of the service routines;

a clustering mechanism devised to group frequently used service routines having mutual referring program jumps into a plurality of program functions ~~in a program function~~ dependent on a predetermined frequency value.

30. (currently amended) The interpreter as recited in claim 29, wherein said statistics mechanism is devised to register the frequency of service routines executed after establishing at least one of said plurality of program functions ~~function~~.

31. (currently amended) The interpreter as recited in claim 29, further comprising an encoding mechanism devised to assign a frequently used service routine a shorter code than service routines executed after ~~a~~ at least one of said plurality of program functions ~~function~~.

32. (original) A method for an interpreter performing operations of computer program code instructions by means of service routines, comprising the steps of:

registering the frequency of execution and the execution parameters of the service routines;

grouping frequently used service routines having mutual referring program jumps in a program function dependent on a predetermined frequency value.

33. (currently amended) The ~~interpreter~~ method as recited in claim 32, further comprising the step of registering the frequency of service routines executed after establishing said function.

34. (currently amended) The ~~interpreter~~ method as recited in claim 33, further comprising the step of assigning ~~an encoding mechanism devised to assign~~ a frequently used service routine a shorter code than service routines executed after ~~a~~ said function.

35. (canceled)

36. (currently amended) An interpreter for executing a computer program using service routines wherein the interpreter ~~A computer program product that comprises an algorithm for interpreting service routines to be executed, wherein the algorithm~~ is configured to:

(a) keep track of how frequently each service routine is executed during execution;

(b) group a plurality of said ~~frequently used~~ ~~and related~~ service routines into one of a plurality of functions based on said frequency of service routine execution; and

(c) dynamically regroup during execution one or more service routines into one of said plurality of functions based on said frequency of service routine execution of said service routines~~assign a service routine code to each frequently used service routine that is shorter than at least one other service routine~~.

37. (currently amended) The ~~computer program product~~interpreter of claim 36 that is further configured to group a first plurality of said service routines in a first one of said plurality of functions and to group a second plurality of said service routines in a second one of said plurality of functions and assign a service routine code to each said frequently used executed service routine grouped in one of said plurality of functions that is shorter than at least one other said service routine that is less frequently executed and that is not grouped in said one of said plurality of functions ~~wherein the algorithm is configured such that the plurality of frequently used and related service routines that are grouped together in one of a plurality of functions are related by how frequently they are executed~~.

38. (currently amended) The ~~computer program product~~interpreter of claim 37 wherein (i) each service routine of said first plurality of said service routines in said first one of said plurality of functions branches to every other service routine of said first one of said plurality of functions and each service routine of said second plurality of said service routines in said second one of said plurality of functions branches to every other service routine of said second one of said plurality of functions, and (ii) each one of said plurality of functions branches to every other one of said plurality of functions~~the algorithm is configured such that the plurality of frequently used and related service routines that are grouped together in one of a plurality of functions are further related as being service routines that are executed in sequence~~.

39. (currently amended) The ~~computer program product~~interpreter of claim 36 wherein the ~~algorithm~~ interpreter is configured to keep track of how frequently each service routine is executed by collecting and recording statistics of how often service routines are executed and what parameters they had.

40. (currently amended) The ~~computer program product~~interpreter of claim 36 wherein the ~~algorithm~~ interpreter is configured to update one of said plurality of functions with service routines that simulate an instruction set~~group a plurality of frequently used and related service routines into one of a plurality of functions according to program jumps between frequently used service routines with regard to a predetermined frequency value~~.

41. (currently amended) The ~~computer program product~~interpreter of claim ~~36~~ 37 wherein the ~~algorithm~~ interpreter is configured to arrange assigned service routine codes of said service routines in one of said plurality of functions in a sequential block~~assign a service routine code to each frequently used service routine that is shorter than at least one other service routine by encoding frequently used service routines grouped into a function with a shorter service routine code than service routine codes of service routines executed thereafter such that frequently used service routines have a faster execution speed~~.